Attorney's Docket No.: 10559/408001 / P10345

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: SPEECH RECOGNITION FRAMEWORK

APPLICANT: SHUVRANSHU POKHARIYAL, SHIRISH AUNDHE AND THOMAS HERNANDEZ

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL584937438US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

April 27, 2001
Date of Deposit

Signature

Daniel Badilla
Typed or Printed Name of Person Signing Certificate

SPEECH RECOGNITION FRAMEWORK

TECHNICAL FIELD

This invention relates to a framework for speech recognition.

BACKGROUND

Sound enabled computers are equipped with a microphone and sound processing equipment, such as a soundcard, that allow the computers to receive sound input. Speech engine software may be installed on sound enabled computers to allow the computers to recognize speech contained within the sound input. The speech typically consists of words or phrases spoken by a user.

Speech-enabled applications, such as dictation software, are equipped to receive speech as input from a user, for example, through a speech engine. The applications typically receive the input through an application-programming interface (``API'') of the speech engine. All communication between the speech-enabled application and the speech engine must conform to a syntax that is specified by the speech API.

Different speech engines may have different APIs. Different versions of the same speech-enabled application are sometimes provided for the different APIs. Alternatively, some complex speech-enabled applications are equipped to communicate with more than one speech API.

DESCRIPTION OF DRAWINGS

FIG. 1 shows a computer that has speech service software installed;

FIG. 2 is a flow chart of the startup process for the speech service software;

FIG. 3 is a flow chart of the startup process for a speech-enabled application;

FIG. 4 is a flow chart showing the processing of speech input;

FIG. 5 shows a speech handler file for a speech enabled application; and

FIG. 6 is a flow chart of the process performed when a user changes focus from one application to another.

DETAILED DESCRIPTION

As shown in FIG. 1, a computer 10 includes a microphone 12 and a sound card 14, which allow the computer 10 to receive sound input from a user operating the computer. The computer also includes a processor 16 and a storage subsystem 18. Storage subsystem 18 is a computer readable medium, such as a, computer memory, a DVDROM, a DVDRAM, a CDROM, a floppy disk, an optical disk, a hard disk, or a hard disk array.

Storage subsystem 18 stores computer software that is executed by processor 16. The computer software includes an operating system 20, a speech engine 22 that recognizes spoken phrases in the sound input, a speech service 24, and two

speech-enabled applications 26 and 28. The operating system is typically a Windows™ operating system by Microsoft Inc., although any other operating system such as MACOS™ by Apple Inc. or a UNIX-like operating system (e.g. Linux, AIX™ or Solaris™) may be used instead.

As will be described in greater detail below, the speech service 24 receives recognized phrases from the speech engine 22 and directs them to the relevant one of the speech-enabled applications 26, 28. Speech service 24 selects a handler function that responds to the recognized phrase based on a speech handler file 30, 32 associated with the speech-enabled application 26, 28. Speech service 24 also initializes and configures speech engine 22 based on the speech handler files 30, 32. The speech handler files 30, 32 are described in greater detail below with reference to FIG. 5.

Thus speech enabled applications 26, 28 interact with the speech engine 22 through the speech service 24, which is configured to communicate with the API 54 of the speech engine 22. As will be described below, the speech service 24 uses the same speech handler file 30, 32 irrespective of the API 54 of the speech engine 22. Consequently, the speech service 24 allows a single version of a speech-enabled application 26, 28 to be used with any speech API that is supported by the speech service 24.

Operating system 20 includes a sound card driver 40 that provides sound information to the software on the computer 10. The operating system also

includes a system registry 42 that contains information on the configuration of the computer 10 and the software installed on it. Certain operating systems may store the configuration information in configuration files instead. The operating system 20 provides an operating system API 44 through which other programs can interact and communicate with the operating system 20.

The user interacts with the computer 10 through a user interface 46 provided by the operating system. User interface 46 is typically a graphical user interface, such as the windows user interface or the X-windows graphical interface, although the speech service may be used in systems with other user interfaces, such as a text user interface or a speech-driven user interface. A user selects a particular application or computer program as the target or ``focus'' of user input through the user interface 46. The application in focus may, for example, be selected by clicking on an icon associated with the program, by typing a sequence of keys on a keyboard, or saying certain keywords.

Speech engine 22 recognizes spoken phrases based on a grammar provided by the speech service 24. Speech engine 22 comprises code modules that perform the functions described below. Speech engine 22 includes a grammar compiler 50, which compiles the grammar into a binary format that is easily loaded into the speech engine 22. Speech engine 22 also includes a speech recognizer 52 that recognizes spoken phrases in sound information from the

sound card driver 40 based on the grammar that has been loaded onto the speech engine. Other software can interact with speech engine 22 through a speech API 54. The API 54 may, for example, be a speech application-programming interface (``SAPI'') developed by Microsoft Inc., an advanced speech API (``ASAPI'') by AT&T Corporation, or a JAVA™ speech application-programming interface (``JSAPI'') by Sun Microsystems.

The first speech-enabled application 26 is a speech-enabled video game. A user plays the video game using speech commands such as ``jump'' or ``kick the monster in the stomach''. The user also performs other functions, such as sending mail to the technical support department of the vendor of the application, using audio commands. A user may, for example, say ``send mail to technical support'' to prepare a letter that is addressed to technical support. Speech-enabled application 26 is stored in a game directory 60 within storage subsystem 18. The game directory 60 also includes the speech handler file 30 of the application 26.

Handler functions 62, which are contained within the first speech enabled application 26, are executed whenever a phrase recognized by the speech engine 22 is directed at the application 26, as described in greater detail below.

The second speech-enabled application 28 is a text editor. A user performs functions associated with entries in the menus of the text editor by audibly stating words corresponding to the

functions. The user may, for example, say ``save file'' to save a file, ``open file'' to open a file, or ``format in bold face'' to format text in bold face font. A user also creates the body of a document by speaking the words he would like to include in the body. The user may, for example, begin a letter by saying ``Dear Mother. How are you?'' The computer 10 recognizes the spoken sounds and provides the text to the text editor as input.

Handler functions 64, contained within the second speech enabled application 28 are executed whenever the speech engine 22 recognizes a phrase that is directed at the second speech enabled application. Unlike the first speech-enabled application 26, the second speech-enabled application 28 has its speech handler file 32 embedded within the executable file of the application 32 as a resource file. The speech service 24 is configured to retrieve the speech handler file 30 from the application 28, as will be described below.

Speech service 24 is a ``service'' or a ``daemon'' that does not does not interact with the user through the user interface 46 and is, typically, executed contemporaneously with other programs on the computer 10. The speech service 24 is automatically started during the startup process of computer 10. Speech service 24 comprises code modules that perform the functions described below.

Speech service 24 includes a speech engine launcher 70 that launches the speech engine 22 when the speech service 24 is first started. All

speech-enabled applications 26, 28 configured to use the speech service 24 that are later executed on the computer 10 use the same instance of the speech engine 22 through the speech server. Thus the speech service 24 eliminates additional processing that would otherwise be required to launch the speech engine 22 every time a speech application 26 is executed.

Speech service 24 also includes a speech handler file loader 72 that loads speech handler files 30, 32 and a grammar generator 74 that generates a grammar file 76 and handling function information 77a from each speech handler file 30, 32. The grammar generator 74, for example, generates the grammar file 76 as described in U.S. Patent Application Serial No. 09/752,994, titled "Specifying Arbitrary Words In Rule-Based Grammars." The handling function information 77a relates different spoken phrases with corresponding functions that respond to the phrases. Each grammar file 76 informs the speech engine 22 of the different phrases to which a speech-enabled application 26, 28 responds. The grammar is typically a context-free grammar with wild card support, although any grammar supported by the speech engine 22 may be used instead. A sample grammar file is attached as appendix A.

The grammar generator 74 also causes the grammar compiler 50 of the speech engine 22 to compile each grammar file 76, producing compiled grammars 78. Compiled grammars 78 are stored within storage subsystem 18, preferably within a rapidly accessible memory,

from where they are later retrieved, as described in greater below. The speech service 24 also includes a default grammar 81 that is loaded into the speech engine 22 when the speech service is first launched.

Speech service 24 further includes a focus module 79, which is notified by the operating system 20 whenever a user changes focus from one application to another. Focus module 79 keeps track of the application that has focus. Speech service 24 includes a phrase parser 84. Speech engine 22 notifies the phrase parser 84 of any phrases in the sound input that are recognized by the speech engine. As will be described in greater detail below, the phrase parser 84 parses the recognized phrases to determine any arguments that may be required by the handler function that corresponds to the phrase. A function caller 86 calls the handler function 62, 64 with the arguments determined by the phrase parser 86. The function caller calls the handler function 62, 64 using an API, such as remote procedure call (RPC), or the component object model (COM) by Microsoft Inc.

As shown in FIG. 2, upon launching the speech service 24, the speech engine launcher 70 queries (200) the operating system 20 to determine whether there is a speech engine 22 installed on the computer 10. The speech engine launcher 70 may query (200) the operating system 20 through the operating system API 44, the system registry 42, or by checking operating system configuration files. The speech engine launcher 70 then queries

(202) the operating system 20 to check if the speech engine 22 is running. If the speech engine 22 is not running, the speech engine launcher 70 starts (204) the speech engine 22.

The grammar generator 74 directs (206) the grammar compiler 50 to compile the default grammar. The grammar generator 74 stores (208) the compiled grammar within storage subsystem 18, preferably in a rapidly accessible memory, and adds (210) the default grammar onto a list of available grammars. Grammar loader 80 loads (212) the compiled grammar onto the speech engine 22. Focus module 79 directs (214) the operating system 20 to notify the speech service 24 whenever the user interface 46 changes focus from one application to another. Focus module 79 also directs (216) the operating system 20 to notify the speech service 24 whenever the execution of a new application is initiated.

As shown in FIG. 3, when a user initiates (300) the execution of an application, for example, by clicking on an icon associated with the application, the operating system 20 notifies (302) the focus module 79. The speech handler file loader 72 then checks (304) if a speech handler file 30, 32 is embedded within the executable of the application as a resource file. If a speech handler file is not embedded within the executable, the speech handler file loader 72 checks (306) if a speech handler file is stored within the same directory as the executable. If a speech handler file is not stored within the directory, the speech handler file loader 72 checks (308) if

speech handler information associated with the application is stored within the system registry 42. If the speech handler information is not stored within the registry, the speech handler file loader 72 terminates the process.

The focus module 79 records (310) identity information associated with the initiated application to indicate that the application has focus. The grammar generator 74 generates (312) a grammar from the handler information extracted from either the speech handler file 30, 32 or the registry 42. The grammar generator 74 then directs (314) the speech engine 22 to compile the generated grammar into a compiled grammar 78a, which the grammar generator 74 stores (316) within storage subsystem 18 and associates (318) with the application. Grammar generator 74 also generates and stores (320) handling function information 77a from the handler information. The handling function information associates spoken phrases with handler functions 62, 64 that respond to them.

Grammar loader 80 unloads (322) the grammar on the speech engine 22 and loads (324) the compiled grammar 78a onto the speech engine. Subsequently, all speech input is recognized by the speech engine 22 based on the compiled grammar 78a corresponding to the initiated application. By loading the grammar 78a that is tailored to the application, the speech service 24 causes speech input to be more accurately recognized by the speech engine 22.

As shown in FIG. 4, when the user speaks (402) into microphone 12, speech engine

22 recognizes (404) phrases contained within the speech and notifies (406) the speech service 24 of the recognized phrase. The speech service 24 identifies (408) the application that has focus by reading identification information previously recorded by the focus module 79. The speech service 24 retrieves (410) handling function information 77a of the application that is in focus and selects (412) a handler function from the information corresponding to the recognized phrase. The phrase parser 84 then parses (414) the phrase to determine any parameter required to call the selected function, as described below with reference to FIG. 5, and then the function caller 86 calls (416) the function.

As shown in figure 5, a speech handler file 90 corresponding to a speech-enabled application 26 (FIG. 1) includes phrases 92 to which the application 26 responds and corresponding handler functions 93 that are to be invoked whenever a corresponding phrase is matched.

The phrases 92 include a simple phrase 92a that consists of a single word ``jump.'' The handler function 93a corresponding to the simple phrase 92a is called without any arguments whenever the simple phrase 92a is recognized.

The phrases 92 also include a wildcard phrase 92b that consists of a sub-phrase 94 (i.e., ``sendmail to'') and a wildcard portion 96 (i.e., *recipient). The speech engine 22 matches the wildcard phrase 92b with any collection of words that begins with the sub-phrase 94. The handler function 93b associated with the wildcard

phrase 92b must be called with an argument 98b named ``recipient''. The name of the argument 98b also occurs in the wildcard portion 96 of the wildcard phrase 92b.

As will be described below, speech engine 22 recognizes a collection of words that matches the wildcard phrase 92b, the phrase parser 84 parses the matching collection of words, extracting any words after the sub-phrase 94 into a variable named ``recipient''. The speech service then calls the corresponding handler function 93b with the ``recipient'' variable as the sole argument.

Phrases 92 further include a complex phrase 92c comprising a first part ``kick'', a wild card for a variable named ``person'' a second part ``in'' and a wildcard for a variable named ``bodypart.'' The complex phrase 92c is matched by any collection of spoken words that has both the words ``kick'' and ``in,'' in that order. The handling function 93c associated with the complex phrase 92c must be called with the variables named ``person'' and ``bodypart.''

When the speech engine 22 recognizes any collection of spoken words that match the phrase 92c, the phrase parser 84 parses the collection of words and assigns the words between ``kick'' and ``in'' to a variable named ``person''. The phrase parser 84 also assigns all words after ``in'' to a variable named ``bodypart''. The function caller 86 then calls the associated handler function 93c with the variables ``person'' and ``bodypart.''

Since the speech handler file 90 does not contain any information that is specific to a particular speech API, the speech engine 24 allows a simple speech application to be used with any speech API version that is compatible with the speech service. The instructions for communicating with the different APIs are embedded in the speech service.

As shown in FIG. 6, when a user changes (600) focus from the first application 26 to the second application 28, the operating system 20 notifies (602) the focus module 79 of the change. The grammar loader 80 then unloads (604) the grammar corresponding to the first application from the speech engine 22 and loads (606) the grammar corresponding to the second application. By loading a grammar that is tailored to the application in focus, the speech service 24 allows the speech engine 22 to recognize phrases in the spoken input more accurately.

Other implementations are within the scope of the following claims.

For example, the speech service may use other means to detect the application that has the focus of the user interface, instead of registering with the operating system to receive notification of a change of focus from one application to another. The speech service may periodically poll the operating system, for example every half-second, to determine whether there has been a change of focus. The polling method may also be used to determine whether the execution of a new

application has been initiated.

Alternatively, the speech service may use the compiled grammars to determine the application that is in focus. In this implementation, the speech service combines all the compiled grammars into a composite grammar, which it loads onto the speech engine. When the speech engine recognizes a phrase from the composite grammar, the speech engine notifies the speech service. The speech service in turn parses the recognized phrase and determines which application has a handler function that responds to the recognized phrase. The speech service infers that the determined application is the application that is in focus.